TECHNISCHE
UNIVERSITÄT
DARMSTADT

Bachelor's Thesis

Pose Estimation and Loop Closing from Video Data

Stepan Konrad
November 2014

Technische Universität Darmstadt
Department of Computer Science
Graphics, Capture and Massively Parallel Computing

Supervisors: Prof. Dr.-Ing. Michael Goesele
M.Sc. Simon Fuhrmann

Erklärung

Hiermit versichere ich die vorliegende Bachelorarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 05.11.2014

Abstract

In robotics the simultaneous localisation and mapping (SLAM) algorithms are a well studied approach to estimate the position of a robot vehicle while creating a map of the surrounding. The majority of these algorithms use odometry or GPS sensors to cope with large outdoor trajectories. From a similar point of view the computer vision community uses structure from motion (SfM) algorithms to estimate accurate camera poses of an unconstrained image data set. In the past few years the video resolution of consumer cameras has reached a level where it becomes attractive for research purposes as input to these algorithms.

The goal of this thesis is to adapt an SfM approach to use this video data. However there are two main problems: The approach has to handle a large number of input frames efficiently while still detecting similar previously seen locations (loops) of the input data without performing an exhaustive matching of all image pairs. This thesis presents an approach using a vocabulary tree guided matching scheme which solves this problem. Performance is compared to exhaustive matching on different input scenes.

However, this is still not sufficient to reconstruct large datasets that contain loop closures in the camera path. Due to the incremental manner of the majority of SfM algorithms, drifts occur during the estimation of camera poses. In this thesis different solutions to this problems are discussed. One specific solution using a global bundle adjustment with additional loop closing constraints is demonstrated on a large outdoor scene containing multiple loops.

Zusammenfassung

Simultaneous Localisation and Mapping (SLAM) ist ein gründlich untersuchtes Forschungsfeld der Robotik um gleichzeitig die Position eines Roboters zu bestimmen und die Umgebung zu kartographieren. Die Mehrheit dieser Algorithmen nutzt zusätzlich GPS oder Odometriedaten um längere Außenaufnahmen verarbeiten zu können. Von einem ähnlichen Standpunkt aus wird in der Computer Vision Gemeinschaft ein Algorithmus namens Structure from Motion (SfM) benutzt, um akkurate Kamerapositionen von einem beliebigen Bilddatensatz zu estimieren. In den letzten Jahren wurde die Videoauflösung von handelsüblichen Kameras so gut, dass diese Videodaten als Eingabe für diese Algorithmen genutzt werden konnten.

Das Ziel dieser Arbeit ist es, ein SfM Verfahren so anzupassen, dass es diese Videodaten verarbeiten kann. Dabei entstehen zwei Probleme: Der Ansatz muss eine große Anzahl von Eingabebildern effizient verarbeiten können und trotzdem ähnliche, bereits gesehene Orte in den Eingabedaten erkennen können. Dies muss möglich sein, ohne ein vollständiges Vergleichen (Exhaustive Matching) aller Bildpaare durchzuführen. Diese Arbeit prästentiert einen Ansatz, der das Matching mit einem sog. Vocabulary Tree beschleunigt. Die Geschwindigkeitsvorteil gegenüber dem Exhaustive Matching wird an verschiedenen Szenen präsentiert.

Dies allein reicht jedoch nicht aus, um große Datensätze, die Schleifen in dem Kamerapfad enthalten, rekonstruieren zu können. Da die Mehrheit der SfM Algorithmen inkrementell arbeitet, treten Drifts während der Rekonstruktion der Kamerapositionen auf. In dieser Arbeit werden verschiedene Lösungsansätze vorgestellt. Eine Lösung, die eine Bündelblockausgleichung mit zusätzlichen Nebenbedingungen zum Schließen von Schleifen nutzt, wird an einem großen außen aufgenommenen Datensatz mit mehreren Schleifen demonstriert.

Contents

1 Introduction

Reconstructing the camera pose—the position and angle the input image was taken from—as well as the 3D geometry of the object that is visible in the image is a research topic of computer vision. A well-known technique addressing this problem is called Structure from Motion (SfM) which yields camera poses with high accuracy on unconstrained input data. Furthermore in the past few years the video resolution of consumer cameras has reached an attractive level to use video data as input to these algorithms. However, in general there are two main problems which require further studies.

First, capturing scenes or landmarks using a video generates far more frames than using a still camera. Utilizing more frames can be a benefit in an SfM process as the chance that frames that would have a high contribution to the reconstruction are missing is significantly lowered. However, as the number of input frames grows, the reconstruction will slow down. In particular the feature matching has a quadratic runtime relating to the number of input images since typically all pairs of images have to be matched. Fortunately on this kind of data, some assumptions can be made to speed up the matching process. Nearby frames usually depict the same part of the scene and thus there likely will be a large number of matches. On large scenes it can be sufficient to match only these nearby frames. This however introduces a new problem: Possible loop closures in a camera path have to be robustly detected. A matching matrix showing a large sparse scene is shown on the right of Figure 1.1.

The second problem is to reconstruct a globally consistent sparse point cloud of the scene as well as a consistent camera path. Especially on large scenes with a low connectivity, for example a camera mounted on a car driving around blocks of houses, a globally consistent reconstruction can be a hard problem, which can be seen on the left of Figure 1.2. In an incremental SfM approach where camera poses and sparse 3D points get reconstructed successively, small errors accumulate and lead to drifts in the camera path. As a consequence, the 3D points get reconstructed erroneously as well. As the reconstruction of a camera path approaches the closure of a loop, both ends get bent due to drift and thus do often not align at all. As the last unreconstructed cameras

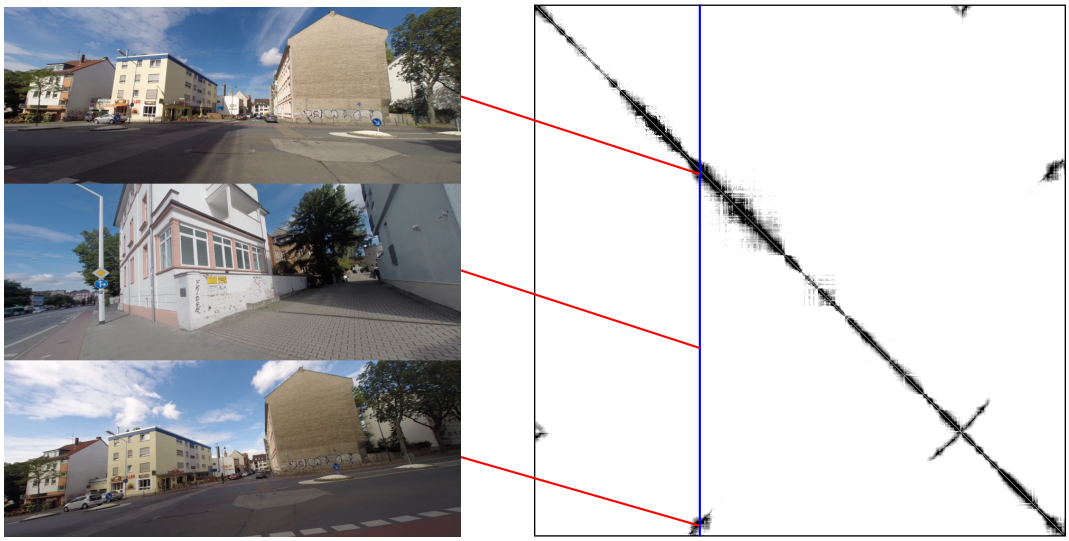

Figure 1.1: *Right:* A sparse matching matrix. Each column corresponds to one image (blue) and visualizes the number of common features shared with each other image (red). *Left: The number of common features is high, when the images depict the same part of the scene.*

Figure 1.2: The overview of a sparsely reconstructed scene including a camera path. Even after a successful image matching, there are two misalignments of the camera trajectory after the pose estimation (red). A loop closure optimization is needed to correct for these errors.

are added to the camera path, the following problem arises: The pose of a camera is estimated using several sets of randomly selected 3D features, that were already reconstructed and correspond to 2D features in the image. The putative pose is then accepted or declined depending on the re-projection error of all corresponding 3D features. This random sample consensus (RANSAC) is a common approach to filter away outliers—for example wrong matches or erroneously triangulated 3D features. However, this process will choose either end of the loop as the foundation for the pose estimation and consider the features at the other end of the loop as outliers. This outlier filtering is crucial because otherwise this would lead to large inconsistencies which disallow the addition of further views. A global bundle adjustment then cannot correct for these errors, as the constraints linking both ends of the closure are thrown away in this outlier removal process.

A promising approach to tackle the matching problem would only start the matching process for pairs that are predicted to match. Ideally the matching time then scales with the number of pairs that actually show a common part of the scenes. This fraction or *density* of the scene can be for instance only 5% of all image pairs. A fast matching approach using Vocabulary Tree Indexing was implemented and will be described in chapter 4.

However, there is no straightforward solution to prevent the drift errors during the reconstruction. Different approaches to optimize erroneous poses are presented in chapter 5. One of this approaches tries to correct alignment errors in the sparse geometry using a global optimization. The results on a real world data set using loop closure constraints are presented in chapter 6.

2 Related Work

2.1 Structure from Motion on Video Data

Using video cameras for pose estimation is a more common scenario in the area of Simultaneous Localization and Mapping (SLAM). Estimating the position of a robot that has a camera mounted at its top is an important and well researched topic in robotic science.

2.1.1 Types of Reconstruction Algorithms

Some SLAM-based approaches use simple camera tracking algorithms. To cope with low-frequency drift most of them use GPS or an Inertial Navigation System [Akbarzadeh et al., 2006]. One intrinsic property of SLAM is that the uncertainty of camera poses and landmark positions is modeled in every step. Moreover there are more advanced approaches that tackle the pose estimation problem on large outdoor scenes including loop closures, for instance LSD-SLAM [Engel et al., 2014]. To correct for loop closures, internally the graph optimization framework g^{2}o [Kummerle et al., 2011] is used.

The Structure from Motion algorithms can be categorized in two different approaches to build up the 3D geometry and reconstruct the camera poses.

Global SfM systems estimate two view poses between pairs of input images and then combine all the local information in a global optimization to achieve a consistent reconstruction. A key property of global SfM algorithms is that the camera pose estimation does not suffer from drift errors. Arising errors are evenly distributed over the dataset and therefore the reconstructed geometry is globally consistent. Moulon et al. [2013] first estimate global rotations using local two view rotations, then relative translations are estimated. Afterwards the global rotations and translations are combined to estimate the final structure and motion of the input scene. A challenging topic using a global approach is the robust removal of outliers in the relative rotations and translations, because it is difficult to reason about relative pose information being outliers or not before the reconstruction process or without having a partial but consistent global

model. Wilson and Snavely [2014] tackle this problem by reducing this difficult problem to a set of one dimensional subproblems by projecting the translations to several planes. This problem can then be solved easier. Secondly they show an improved solution to the translation problem—to find global translations using relative ones as input. In addition they not only use camera-to-camera constraints but also camera-to-point constraints to robustify their optimization.

In *incremental* SfM systems, the process starts with a small set of two or three input images to reconstruct a starting set of 3D points. The remaining cameras are then incrementally added to the reconstruction and as more images are added, more 3D points are triangulated. In contrast to the problematic outlier removal of global SfM systems, the situations here is easier as the incrementally reconstructed cameras are aligned to a globally consistent model. If the estimated pose of a newly reconstructed camera is erroneous, it does not have a huge impact on the reconstruction of the remaining views. However, in an incremental approach the bundle adjustment is a crucial process and must be executed many times to correct drifts and other errors.

The SfM algorithm of Klingner et al. [2013] is used to create a planet-scale point cloud from street-view images. Their approach additionally uses data from GPS and IMU (inertial measurement unit) sensors. To correct drift errors at loop closures, they add relative pose constraints to the bundle adjustment. These relative poses are computed by aligning corresponding 3D point sets using a robust least-squares estimation [Umeyama, 1991] in a RANSAC loop.

2.2 The Loop Closing Problem

One particular problem of these incremental approaches is the accumulation of errors. As new images are added to the bundling process, errors accumulate and lead to drifts in camera and feature positions over time. These errors are caused by wrong feature matches, uncertainties in triangulation and other numerical problems. As a result of these drifts, possible loops of camera position are neither detected nor treated and lead to overlapping 3D geometry or open camera loops.

The previously described scenario is called the Loop Closing Problem. A solution to this problem could be split up in two separate subproblems:

1. Detect possible loops or overlapping camera trajectories and
2. verify and close the detected loops.

A lot of research to address Loop Closing was done in the last decade especially for SLAM-based systems. According to Williams et al. [2009], loop closing approaches can

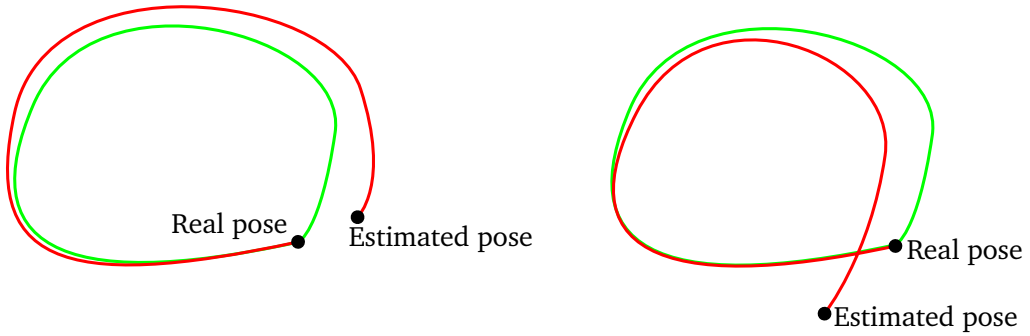

Figure 2.1: Illustration of the Loop Closing Problem. *Left:* After the pose estimation the loop is not closed. *Right:* There is an overlap of the camera path after the pose estimation.

be categorized into the following three categories according to where the data association for detecting the loop closure is done.

2.2.1 Map to Map

To detect a possible loop in a *map to map* approach the reconstructed 3D features are split up in several submaps. Then corresponding features are searched in two selected submaps. The algorithm used by Clemente et al. [2007] is a variation of the original Geometric Compatibility Branch and Bound algorithm. This system computes the relative scale, rotation and translation to align two submaps using the largest compatible set of common features in these two submaps.

2.2.2 Image to Map

Another approach is to search for point correspondences between the recently captured video frame and the global map. Williams et al. [2008] search for a sufficient number of common features to detect a loop closure. For each detection their algorithm uses RANSAC and the three-point-pose algorithm to start the re-localization process: An *alternative* camera pose is estimated for each verified loop closure. Using two or more camera poses as well as the alternative poses, a transformation can be calculated which corrects for the wrong geometry.

2.2.3 Image to Image

Even without regarding the map of 3D features, loop closures can be detected using the visual appearance of the video frames. Cummins and Newman [2008] use a visual

vocabulary requiring offline training which is able to calculate the probability that two images show the same region.

In the literature an increasingly common approach to find image to image correspondences efficiently is a vocabulary tree indexing based approach [Nister and Stewenius, 2006]. Their method, which is inspired by object recognition topics, consists of two steps. First they build a database composed of all input images, then they match a query image against the database. Using this tree the features of each image are quantized to create a single descriptor for each image which allows for very efficient matching of image pairs.

The parameters of the vocabulary tree are the branching factor—the number of children per node—and the maximum depth of the tree. Frahm et al. [2010] claim that using a broader tree yields better results in their experiments.

Scaramuzza et al. [2010] use the vocabulary tree approach to detect loop closures in videos captured by omnidirectional cameras. The loop hypotheses which are gained trough this visual place recognition are then verified using geometric correspondences. Finally the new correspondences between 2D image features and 3D points are added as loop closing constraints to the bundle adjustment optimization. A similar approach using a vocabulary tree as well is presented by Klopschitz et al. [2008].

3 Structure from Motion from Video Data

One main part of this thesis is to adapt a general Structure from Motion algorithm to work efficiently and robustly on a large number of video frames. There are several steps in the SfM pipeline which can be implemented differently or parameters that have to be changed. Therefore the implementation is mainly based on the Multi-View Environment (MVE) `C++` library [Fuhrmann et al., 2014] which already has a broad set of useful algorithms which can be altered easily. During the work on this thesis, some additions were made to MVE.

3.1 Keyframes

A simple way to speed up the overall process would be to only use certain frames from the video. Consumer cameras capture video footage usually in compressed form. One naïve approach to reduce the number of processed frames is to only use the keyframes which were used in the video encoding algorithm. The advantage is that these frames suffer less from video compression artifacts compared to the differential frames.

Additionally frames containing large amounts of blur can be filtered away. Therefore the video frame with the highest gradient magnitude summed over all pixels can be used. Though this approach only works if there are frames with a sufficient high sharpness within the search window.

A more advanced approach is to adaptively select the number of video frames to reconstruct. The pose estimation process could start with using only the video keyframes. Then the following heuristic could influence the overall number of bundled video frames: If there are many feature matches between the last two matched video frames or if the camera pose could be very well explained just using an homography, the algorithm could skip the last frame to speed up the bundling process and to enlarge the baseline for more accurate results. If the next chosen video frame can not be bundled because there are too few correspondences, the algorithm should try to bundle an in-between video frame instead.

3.2 Features

For each input frame, SIFT features [Lowe, 2004] are extracted and descriptors are computed. Usually the descriptors are computed on an image pyramid to achieve scale invariance. Videos usually have a lower resolution than photos and suffer from compression artifacts and motion blur. Some tests have shown that sometimes there are too few geometrically verified features matches. To gain overall more features per image it is very beneficial to scale up the input images. Additionally SURF [Bay et al., 2008] features can be used to get a denser point set.

3.3 Matching

Using generic SfM algorithms, the exhaustive matching between all possible image pairs grows quadratically in the number of input images (n), more precisely there are

$$\frac{n(n-1)}{2} \tag{3.1}$$

image matches. This leads to a drastic computational load because the matching has a runtime in $\mathcal{O}(n^2)$.

3.3.1 Temporal and Spatial Coherence

It is possible to speed up the matching by matching a new video frame only to the k previous frames. Using video frames as input, the temporal coherence that arises from the spatial coherence can be exploited. A sensible value for k has to be evaluated and depends strongly on the distance between captured frames and the field of view of the camera system. As the initial pair requires a wide baseline for accurate reconstruction, it can be necessary to set k to a higher value for the first few frames of the input video.

3.3.2 Match Filtering

Using a ratio test as proposed by Lowe [2004] is a common approach to filter feature matches and lower the false positive rate of correspondences. Instead of using a constant threshold for the feature distances, a feature correspondence is considered correct only if

$$\frac{\|F - N_{1\text{st}}\|}{\|F - N_{2\text{nd}}\|} < t \tag{3.2}$$

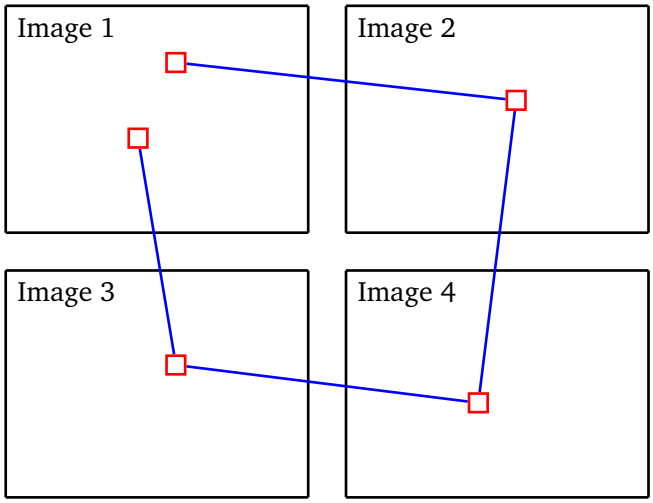

Figure 3.1: A track (blue) that contains an image twice, because two different features (red) belong to this track. This is an implausible conflict and is therefore removed.

holds. F is a feature, $N_{1\text{st}}$ the nearest neighbor and $N_{2\text{nd}}$ the second nearest neighbor. Typically the threshold t is set to 0.7 but using a stricter value, for example $t = 0.5$ yields a more robust matching which both speeds up the matching process and reduces the false positive rate of matches. The higher matching speed is due to the fact that with a lower t more false matches are filtered away and do not have to pass the geometric filtering step.

Geometric filtering of feature matches can be achieved by estimating a fundamental matrix in a RANSAC loop for each putative matching image pair. The matches in the image pair get rejected if the inlier count is less than a specific value.

3.4 Tracks

Corresponding 2D features are tracked in two or more images and are later used to triangulate the track. The number of images sharing the feature is called track length. However tracks can arise that contain one image multiple times. This implausible conflict is demonstrated in Figure 3.1. These tracks have to be removed to prevent wrong triangulations.

3.4.1 Triangulation

In the iterative reconstruction, tracks that belong to newly estimated poses are then triangulated in a least-squares sense. The resulting 3D point is the closest to all camera rays trough the 2D features on the image plane. Tracks are then discarded, if the triangulation angle is too small, the re-projection error is above a threshold or the point appears behind a camera.

3.5 Pose Estimation

3.5.1 Initial Pair

The first step of an incremental SfM bundler is to choose and process an initial pair of images. Choosing an initial pair with a too narrow baseline leads to large depth uncertainties during the triangulation. Therefore the initial pair in a video sequence can be chosen as the first video frame and the next video frame that meets the following requirement: The relative camera pose is not describable by a homography. A useful procedure is to accept the initial pair if the inlier count after the RANSAC step is less than 50%. This ensures a sufficiently wide baseline. Moreover one can enforce a stable triangulation angle. After finding a plausible fundamental matrix and thus a relative alignment of the two cameras, the correspondences of the initial pair are triangulated to create an initial set of 3D features. Intrinsic and extrinsic camera parameters as well as the positions of the 3D points are refined or corrected by running a bundle adjustment optimization.

3.5.2 Remaining Frames

To estimate the global camera poses of the remaining frames, each view is aligned to the existing 3D feature set using 2D-3D-correspondences. Thereby the camera is aligned absolutely towards a global coordinate system.

The 6-point algorithm [Hartley and Dano, 2000] uses six or more correspondences to estimate the camera position as well as the intrinsic camera parameters in a RANSAC loop. If the intrinsic parameters are approximately known, it is more useful to use the Perspective-Three-Point (P3P) algorithm [Kneip et al., 2011] performs more robustly using only three 2D-3D-correspondences. Three samples are drawn from the correspondences using a RANSAC loop to estimate a new camera pose. A fourth correspondence is needed to eliminate the ambiguities. All correspondences are re-projected using this

possible camera poses to count the number of inliers until the pose with the highest amount of inliers is used.

Once the RANSAC loop yields a consistent result, the pose is again optimized in a bundle adjustment optimization.

3.6 Bundle Adjustment

The bundle adjustment optimization plays a key role in any Structure from Motion algorithm. The mathematical approach of the optimization is to minimize the re-projection error with respect to the triangulated points as well as the intrinsic and extrinsic camera parameters.

As a consequence of the incremental approach, errors accumulate as new frames are added to the bundling process. To counteract these errors, it is necessary to run a full bundle adjustment after a new camera pose was reconstructed to re-estimate all camera poses as well as focal length and distortion parameters. The position of the already reconstructed 3D points is globally optimized as well. However, to speed up the iterative reconstruction it can be sufficient to run the full global optimization only after a couple of cameras have been added to the reconstruction. Hence after the pose estimation of a new frame, a single camera bundle adjustment can be run, which only optimizes this new camera parameters.

For this thesis, initially Multicore Bundle Adjustment (MBA) [Wu et al., 2011] was used as a library for the bundle adjustment optimization. MBA is a library designed to be very efficient using GPU computation. To have a better control on the bundle adjustment formulation, the bundler was modified to use the more general Ceres Solver [Agarwal and Mierle, 2012]. This is a library for solving non-linear least squares problems which additionally enables robustified bound constraints. Nevertheless Ceres has several optimizations for bundle adjustment problems and is widely used in recent projects. In contrast to MBA, Ceres is more flexible, meaning the error function can easily be adjusted. Additional constraints can be introduced or some residuals can be weighted differently.

The error function that is optimized is usually designed as

$$E_r = \sum_{x_\mathrm{m}} \| f_\mathrm{rd}(x_\mathrm{m}) - K(Rx + t) \|_2^2 \tag{3.3}$$

where x_m are the 2D features—or measurements—that are compared to the rotated (R), translated (t) and projected (K) 3D track positions (x). In this case the radial distor-

tion is applied to the measurements (x_m) instead to the predicted projections which is modeled by the function f_{rd}. Several parameters can be set constant during the optimization, for instance after a single camera is added. For this camera only R, t and K are optimized, whereas the other parameters as well as the track positions (x) stay constant. Experiments have shown that using Ceres solver with the Sparse Schur solver (see [Agarwal and Mierle, 2012]) achieves a performance which is comparable to MBA.

3.7 Problems

The exploitation of the temporal and spatial coherence during the matching leads to the following problem: Loop closures of a camera path in a scene are not detected and thus cannot be aligned towards a consistent model. The consequences of this can be seen on the right of Figure 3.3.

The presented *Cylinder* scene is a synthetically created test set using the 3D graphics software Blender [Blender Foundation, 2014]. The model consists of an opened cylinder as geometry and a camera path with 150 camera poses following a circle around it. A rendering can be seen in Figure 3.2. The camera position of the last frame is beneath the camera position of the first frame to create a loop closure. The cylinder's geometry is bump mapped using noise to reduce the amount of homography inliers for better two-view pose estimation capabilities. Perlin noise [Perlin, 2002] is used to texture the surface of the cylinder which allows for extraction of robust SIFT features.

A solution has to address two subproblems: First, a method has to be found that approximates the result of an exhaustive matching in less time. The second problem is to correct the errors caused by drift.

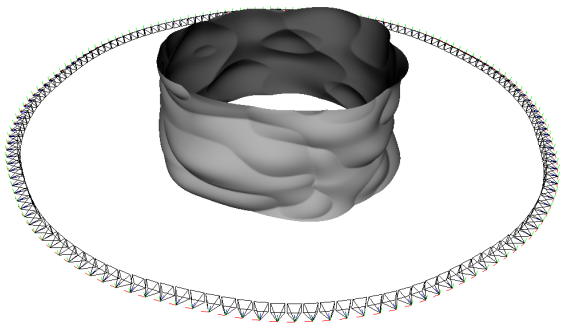

Figure 3.2: A rendering of the synthetic loop closure dataset. An open cylinder with distorted side geometry is surrounded by a camera path consisting of 150 positions arranged in a circle. The cameras are depicted as frusta in the figure. The noise texture is not shown.

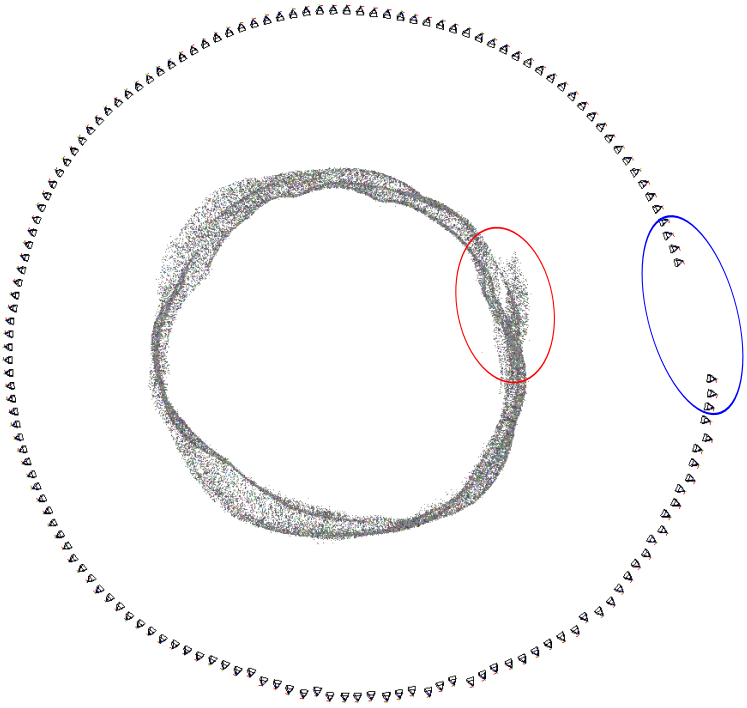

Figure 3.3: The reconstructed *Cylinder* dataset without any loop closing. Towards the end of the loop, camera pose reconstructions were subject to drift errors (blue) and duplicated geometry is clearly visible (red).

4 Loop Detection

Before the loop closing problem can be solved, it is necessary to know which parts of a scene belong together—or in this particular case: which pairs of images show the same part of the scene.

While exhaustive matching of all image pairs is often computationally infeasible, a faster way is to search for possible matches using a heuristic which will be described in the following. Once all matching image pairs are found and all consistent tracks are built, in theory all necessary information to reconstruct the scene and simultaneously close possible loops is available.

4.1 Tree-based Matching

Preliminary tests were made using the Fast Library for Approximate Nearest Neighbours (FLANN) [Muja and Lowe, 2009]. All SIFT features of the scene were inserted into a kd-tree using k-means clustering on each node of the tree. Thus, matching images can be found by searching similar features in all other images. This can be done in a fraction of time in comparison to exhaustive matching.

However this approach yields image matches with a high false positive ratio which can be seen on the left of Figure 4.1. This is due to the approach implicitly using a distance threshold instead of a ratio test to compare image features. A ratio test is more difficult to implement in this situation because two feature matches have to be in the result for each image. The result is compared to an approach using one tree per image which can be seen on the right of Figure 4.1. The similarity matrix is closer to the exhaustive match matrix which can be seen on the right of Figure 4.3. The loop closures are now clearly visible but the benefits of time saving are not big enough to pursue this approach.

In addition a Pyramid Match Kernel approach was tested using the library libpmk [Grauman and Darrell, 2007] which yields similar results at similar runtime compared to FLANN.

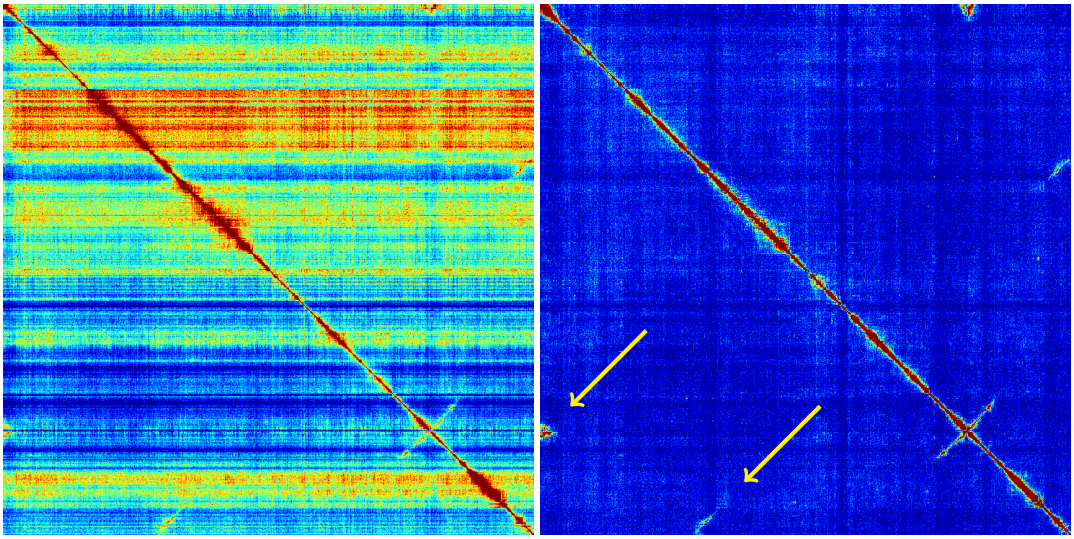

Figure 4.1: Two similarity matrices of the *CityBlocks* dataset using a match prediction based on FLANN. Each row corresponds to each view and shows an image similarity score to all other images. The matrix is jet-colored for better visualization. *Left:* Similarity matrix result using a single tree for *all* images. *Right:* Result of an approach using one tree *per* image. The loop closures are clearly visible (yellow) but the runtime is substantially slower compared to the single tree approach.

4.2 Vocabulary Tree Indexing

The false positive ratio can be improved by using vocabulary tree indexing. The basic idea is to create a single descriptor for each image.— These descriptors can then be compared very efficiently to find image pairs with similar content. Subsequently, these pairs can be used as input to the SIFT feature matching. The creation of this single image descriptor can be accomplished in the following way.

All 128-dimensional SIFT descriptors of all input images are clustered using k-means clustering. The cluster centers can be regarded as child nodes in a tree with k children (branching factor). This is recursively processed until the descriptors fit into a leaf node. The SIFT features are then quantized by assigning each feature the leaf ID it belongs to. The number of SIFT descriptors of one image is then counted for each leaf node to create a single descriptor for each image. This is illustrated in Figure 4.2. The dimensionality of the of this descriptor is the number of leaf nodes of the vocabulary tree, though the descriptor contains many zero-entries and can therefore be compared to others very efficiently.

A crucial step is the weighting of tree nodes according to the frequency of the SIFT descriptors that are contained: Descriptors, that appear in many images receive a low

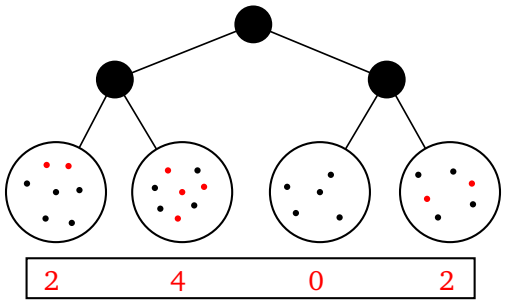

Figure 4.2: Illustration of the Vocabulary Tree Indexing. The SIFT descriptors (small dots) are inserted into a k-ary tree (here $k = 2$). To create a single image descriptor (rectangle), the features that belong to this image (red) are counted for each leaf node.

weight because they are not very discriminative. On the contrary, descriptors that have a high uniqueness receive a higher weight. This is called TF-IDF (term frequency–inverse document frequency) scheme and comes from the topics of information retrieval. The weights of the tree nodes are then applied to the single image descriptors.

As some experiments have shown, a filtering step, which removes SIFT descriptors that belong to features with a small scale, is not necessary and sometimes worsens the results of the scoring. This can be seen in conformity with the results from Nister and Stewenius [2006], stating that the quality of the scoring depends simply on the number of descriptors in the database—the more the better. The implementation of quantization, scoring and matching is based on the library VocabTree2 [Agarwal et al., 2011].

In Figure 4.3 the similarity scores of the vocabulary tree image matching is compared to the exhaustive matching. It is visible that the similarity of an image pair correlates with the number of matches of the exhaustive matching.

4.3 Additional Match Extraction

To approximate the result of an exhaustive feature matching in less time, only those entries—the image pairs—in the similarity matrix can be passed to the matching process that are above a constant threshold. However, it is difficult to find a meaningful absolute threshold because the scoring values vary under different sets of input parameters, for example the branch factor or depth of the vocabulary tree or the number of features or images. The results of varying the branch factor can be seen in Figure 4.4. Different scenes result in different value ranges. A normalization on this data is difficult as well.

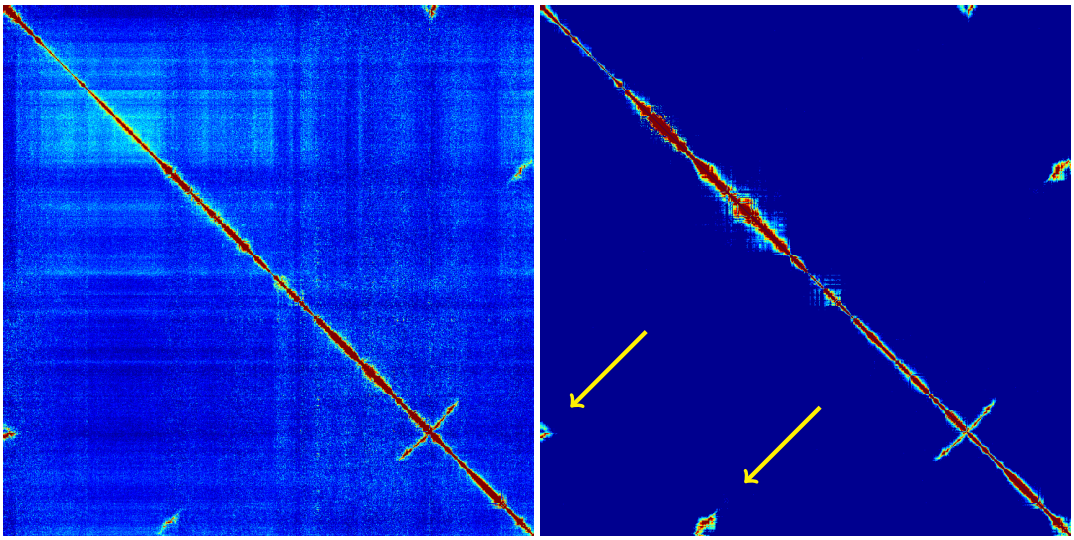

Figure 4.3: The similarity matrix on the *left* is the output of the loop detection phase. Each row and each column corresponds to a single frame. The similarity values on the correlate with the number of feature matches per image pair using exhaustive matching (shown on the *right*). Again the two loop closures (yellow) can be seen in both matrices.

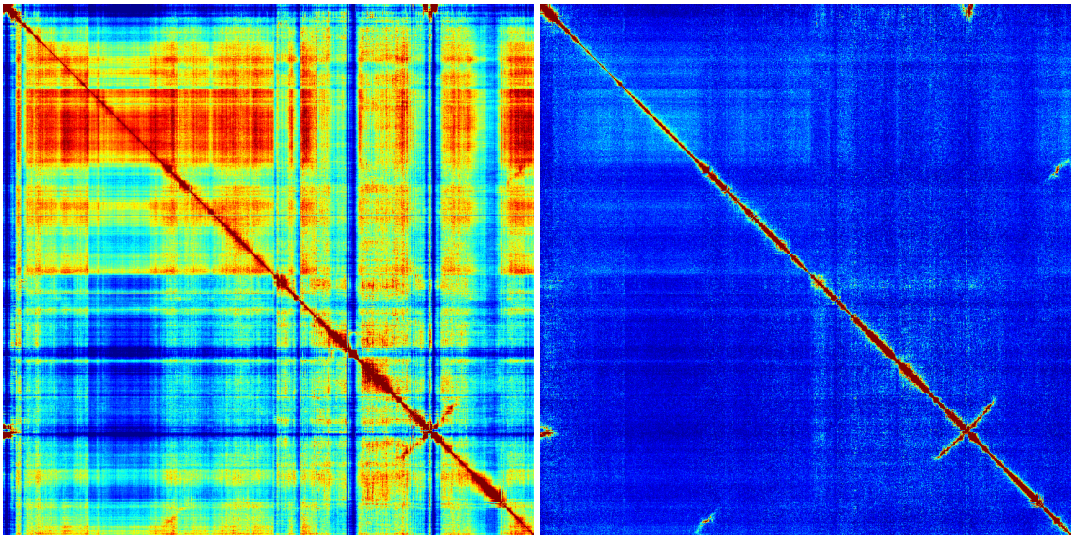

Figure 4.4: Comparison of vocabulary tree results using different branch factors. *Left:* A branch factor of 3 produces false positive matches in regions where the input images contain more repetitive structures. *Right:* Increasing the branch factor to 6 allows for more features being inserted into the vocabulary tree and therefore the result improves substantially. For all trees a depth of 7 has been used.

An solution to this is to sort the image pairs by their similarity value and iteratively match these images from a high to a low similarity value. However, it is more efficient to match larger chunks of k image pairs at the same time. The parameter k can for instance be set to the number of input images. To reduce the chance that image pairs that are neighboring to successfully matches image pairs are missed, additionally the following region growing strategy is used: If an image pair was successfully matches, the direct horizontal and vertical neighbors in the match matrix are matched in the next iteration. This iterative process can be summarized as follows:

1. The k pairs with the highest similarity value from the similarity matrix are collected as well as
2. the pairs from the region growing strategy.
3. The proposed pairs are then matched and geometrically filtered using the standard matching process.
4. If the number of unsuccessful matches after the feature matching is above $0.7k$, the iterative matching is terminated.

Using this technique a match region around the *seed points* from the vocabulary tree approach is grown until the majority of the proposed matches were unsuccessful. This region growing can be seen in Figure 4.5. Lower values than $0.7k$ can be used to exit the iterative matching earlier, but then good image pairs can get lost. However using this value results in a good approximation: On an average, the matching matrix using the vocabulary tree approach is missing only 5% of the matches that would have been produced using exhaustive matching.

In many cases, this leads to a sparse matrix which is diagonally dominant with some additional *spots* or regions at the loop closure points.

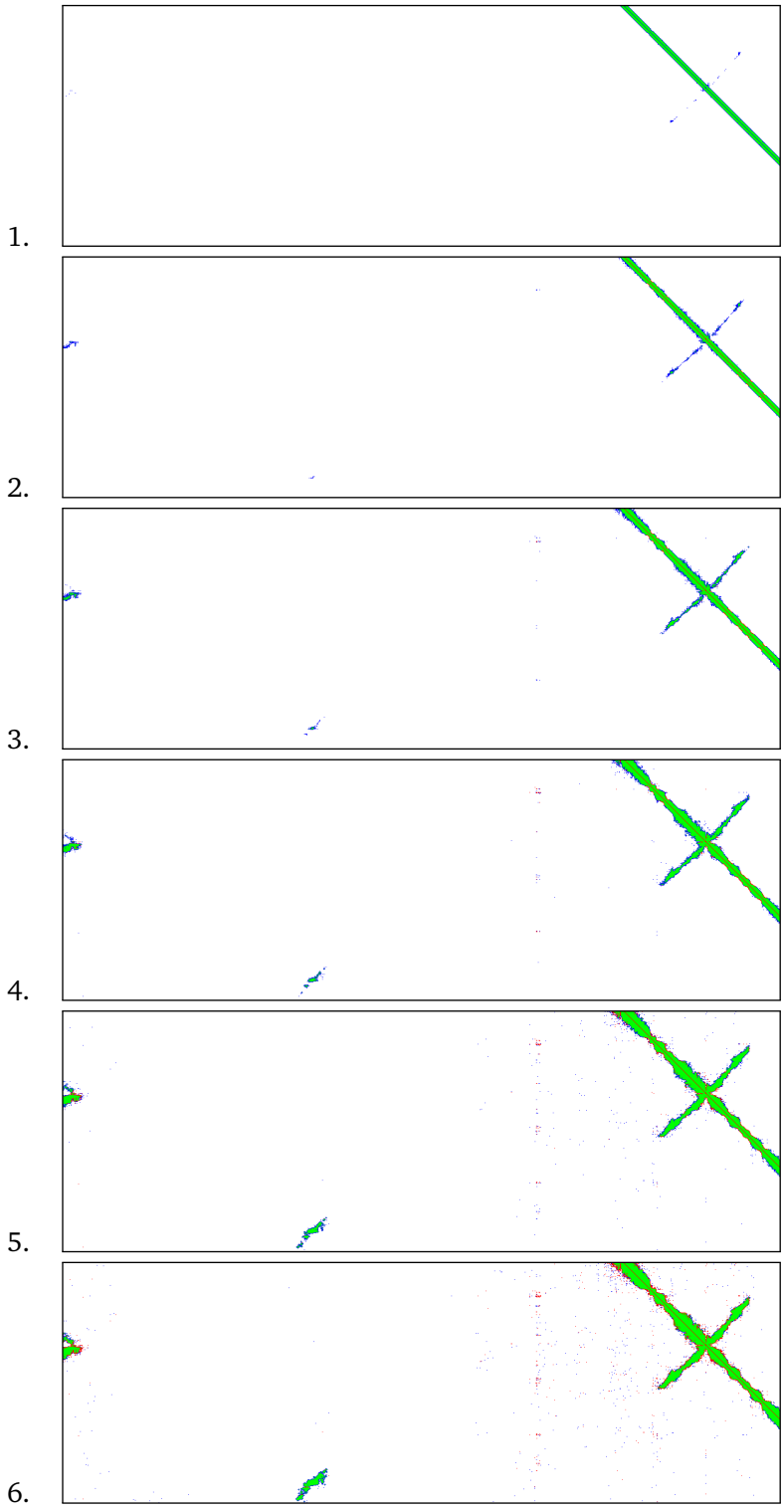

Figure 4.5: The illustration shows six iterations of the match extraction. *Green* points depict successfully matched image pairs. Proposed pairs that will be matched in the next round are marked *blue*. *Red* points show unsuccessful matches and *white* areas have not been matched at all.

5 Loop Closing

Having the matching information, the tracks can be built and the incremental Structure from Motion can start. Using these tracks as input to an incremental pose reconstruction method should deliver a consistent sparse 3D model as well as plausible camera poses. However this works only well on datasets that show a scene with a high connectivity between the views. That could be for instance a single object which was filmed from surrounding camera positions. This kind of datasets results in a denser image matching matrix.

On these datasets the loop detection delivers feature matches that are sufficient to consistently reconstruct a camera path and sparse geometry which can be seen in Figure 5.1. The method may still work on small datasets with a sparse matching matrix. However, it can fail on large sparse datasets with a linear camera path. (i.e. about 1000 input images).

Filming the walk around two city blocks provides such a data set with one or more loop closures. A video sequence was captured using a GoPro HERO3+ Black Edition filming the houses orthogonally towards the walking direction. The dataset is called *CityBlocks* in this thesis. Due to the large radial distortion, an intrinsic camera calibration is needed. This was done using the Omnidirectional Camera Calibration Toolbox for Matlab [Scaramuzza et al., 2006] which supports the calibration of fish-eye lenses. Undistorted images were extracted from the video and stored as MVE scene for further processing. The camera trajectory contains two overlapping camera trajectories as can be seen in Figure 5.2.

The globally inconsistent scene reconstruction and pose estimation has the following cause: As a camera trajectory loop gets reconstructed towards its closure, drifts in the camera path or reconstructed points lead to parts that do not align very well. At the closure of the loop, the RANSAC step for finding the most plausible pose for a new camera is making a decision between two plausible poses. The camera gets reconstructed at the location which led to the higher amount of inliers. The other possible pose will get thrown away, counting the associated points as outliers. This is the critical part of

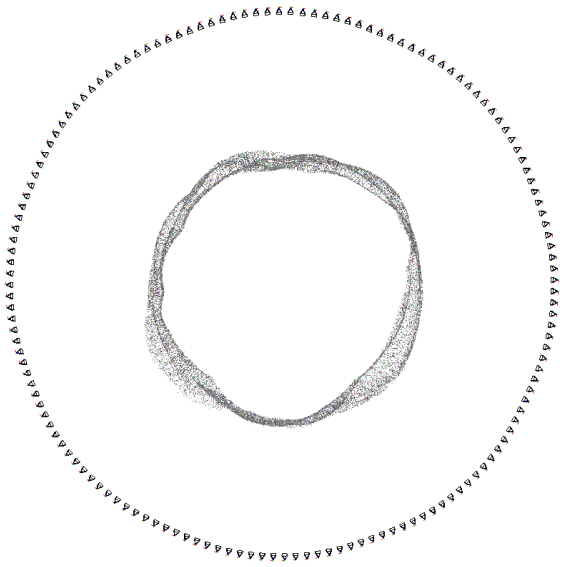

Figure 5.1: The *Cylinder* dataset after a successful loop detection. The small errors which can be seen in Figure 3.3 are now resolved.

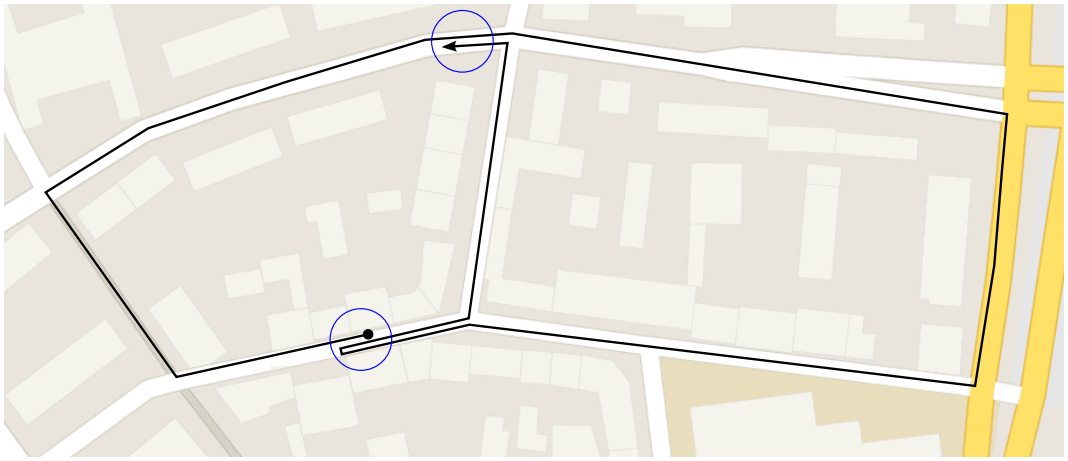

Figure 5.2: A schematic of the large real world loop closing dataset *CityBlocks*. The camera trajectory contains two loop closures marked in blue.

Figure 5.3: Reconstruction of Dataset *CityBlocks* using a standard approach after a successful loop detection. The two parts of the scene that did not align correctly are highlighted in red.

the pipeline that results in erroneously reconstructed scenes which can be seen in Figure 5.3. Notice that the open or overlapping camera trajectories are often at camera positions that were reconstructed later in the incremental bundling.

5.1 Alternate Pose Estimation

One possible approach to solve these errors is to run the pose RANSAC multiple times. In a first round the camera pose is estimated and the inliers are removed from the initial set of images of a track. A second run can then be started to estimate the alternative camera pose using the remaining images of this track. Working with two possible camera poses is somewhat similar to the proposal of Williams et al. [2008] where they use a relocalisation module to estimate two or more poses and then add constraints to their Kalman filter update which is used for the pose estimation.

Here, these constraints can be formulated using a quadratic error between the two camera translations. A global bundle adjustment step can then be used to correct for these errors and optimize the reconstruction towards a globally consistent state. However, it is difficult to figure out, if the estimated alternative pose is correct or plausible just from the ratio of inliers, which can be seen in Figure 5.4.

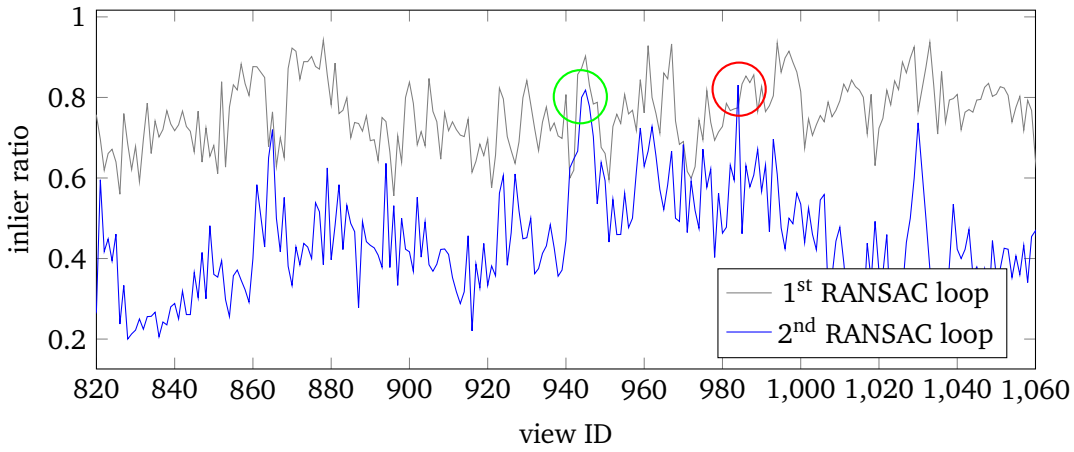

Figure 5.4: Ratio of inliers to all correspondences for the first and second RANSAC loop. There are several view IDs where the second RANSAC yields a plausible result in terms of the inlier ratio, for example at the red circle. However, only at view ID 940 (green circle) there actually is an open loop in the reconstructed camera trajectory.

5.2 Optimization using Track Splitting

An alternative procedure to globally optimize the pose estimation can be accomplished after the whole reconstruction is done. All cameras now have a known global rotation and translation with some errors coming from drifts. During the reconstruction, the information needed to close loops is actually getting lost because the tracks are altered in several ways: Some tracks are removed from the reconstruction if the re-projection error is too high. On the other hand, if a new camera pose is reconstructed, the matches that are considered outliers are removed from the respective tracks. the current camera is then removed from these tracks as well.

Before the pose estimation, the needed information for loop closures is still encoded in the tracked matches. This can now be used to establish loop closure constraints based on the triangulated track positions which can be subsequently used in a bundle adjustment optimization.

Starting from a complete iterative reconstruction with unclosed loops, the initial tracks as computed after the matching are used. The goal of this approach is to re-triangulate this tracks given the already estimated camera poses, while splitting tracks up as it gets necessary: The splitting allows each track to be triangulated at different positions at the same time. This happens at parts of the scene where the camera trajectory is misaligned.

The triangulation and track splitting can be summarized by the following approach which iterates over all tracks:

1. From all cameras that belong to a track, two or three cameras are selected that contribute to a sufficient low re-projection error.
2. This initial set of compatible cameras is grown until no further camera can be added.
3. The track is split up into two tracks—or rather two sets of cameras—the remaining cameras are moved to the new track.
4. The correspondences between this two tracks are then stored in a list for further processing.

This corresponding track list is then filtered to remove tracks that do not triangulate successfully.

An additional distance constraint on the track positions is then added to the bundle adjustment error function: If t_1 and t_2 are the two different triangulations of the split track then the constraint can be formulated as

$$E_t = w \sum_{(t_1,t_2)\in M} \|t_1 - t_2\|_2 \tag{5.1}$$

where M is the list of the splitted tracks and w is a weight. Usually there are only few tracks that contain a plausible loop closure information. Thus it is crucial that the re-projection error for these tracks receives a high weight during the bundle adjustment optimization. Without this weight, the optimization accepts the re-projection error that is increased by merging the split tracks, without correcting the connected camera poses at all.

This approach is capable to correct for camera path and scene misalignments in large outdoor datasets. This is demonstrated in section 6.2.

6 Results

6.1 Increased Matching Speed

The verification of performance benefits was done using various benchmarks. The aim of the implementation using a vocabulary tree was not to reduce the quadratic runtime of the matching process to a linear one. The runtime is still quadratic but the number of image pairs is strongly reduced by the vocabulary tree based image similarity. The similarity calculation itself is quadratic in the number of input images but its runtime is an order of magnitude faster than the feature matching. This is shown in Figure 6.3. All benchmarks were performed on a machine with two Intel®Xeon®CPU E5-2650 v2 processors.

Due to this objective, the best possible runtime of the vocabulary tree guided matching is the runtime of the exhaustive matching times the density of the dataset. Thus in theory this is

$$t_{\mathrm{vt}} = d\,\frac{n(n-1)}{2}\,t_{\mathrm{m}} \tag{6.1}$$

where d is the density and n the number of input images. t_{m} is the time consumed by matching a single image pair.

Summarizing there are two main factors the computational runtime depends on: The overall number of images or image pairs and the density of the matching matrix. Both factors were evaluated independently: From a real word dataset several datasets were derived with changing density. This was done by replicating images (to increase density) and removing parts of the sequence (to keep the number of input images constant). The vocabulary tree approach was then compared to the exhaustive matching and an accelerated exhaustive matching which uses low-resolutions features—determined by their SIFT scale—that are matched first to determine if the subsequent matching is going to be successful [Fuhrmann et al., 2014]. Those runtimes as well as a theoretical lower bound are plotted in Figure 6.1.

In Figure 6.2 it is clearly visible that the timings of vocabulary tree guided matching as well as exhaustive matching have a quadratic shape in relation to the number of input

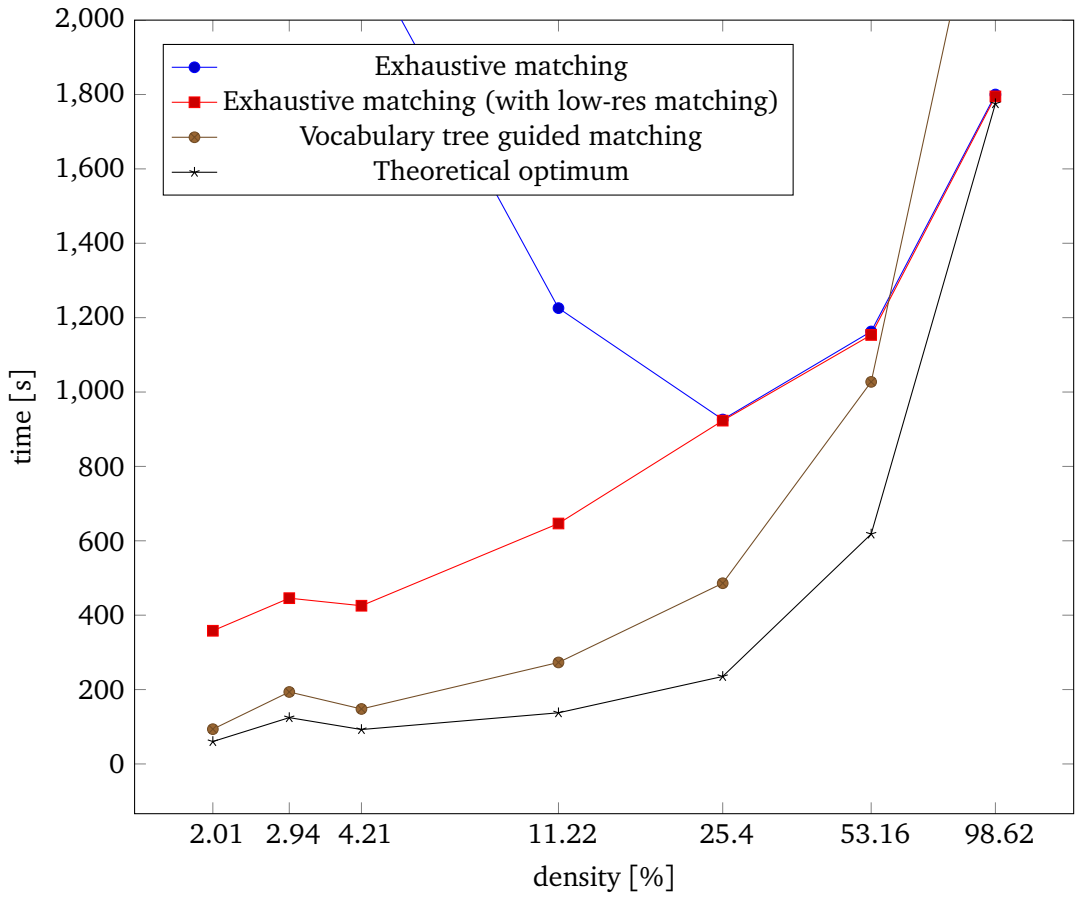

Figure 6.1: Performance comparison between exhaustive matching and the vocabulary tree guided matching on several generated test scenes with different match matrix densities. The theoretical optimum is plotted for comparison which is approximated by multiplying the runtime of exhaustive matching with the density. The benchmark was created using a fixed number of 800 input images. Note that the runtime of exhaustive matching is not constant, possibly due to the geometrical filtering and the varying number of features.

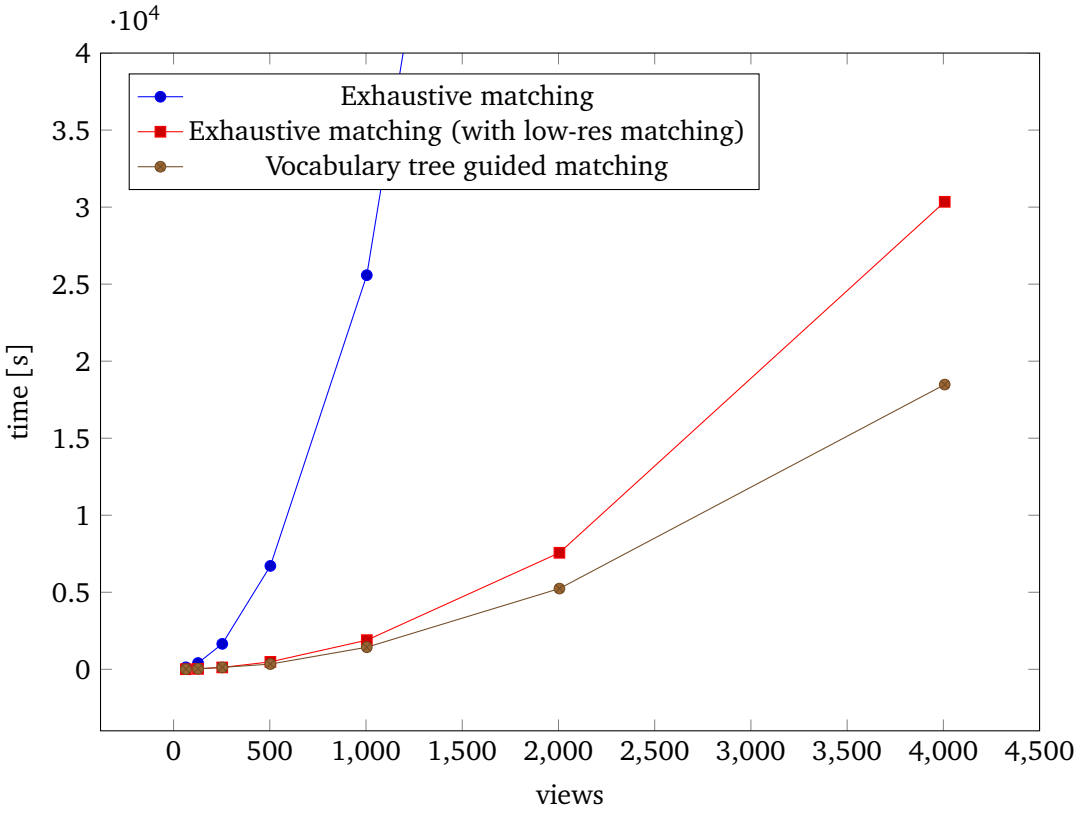

Figure 6.2: Performance comparison of the exhaustive matching (with and without low-res matching) versus the vocabulary tree guided matching on scenes with increasing number of views and a constant density of 4.31%.

views. The guided matching is a factor faster which depends on the sparsity of the scene.

6.2 Track Splitting Results

To verify whether the track splitting works in the way it was meant to be, the *Cylinder* dataset was used in a modified version. By using a fixed wrong focal length, the geometry gets reconstructed erroneously. Figure 6.4 shows a visualization of the tracks splitting that is then used as input to the optimization problem.

As a reliable test, synthetic input data is not appropriate because synthetic scenes can usually be reconstructed without large drifts in the camera path. The track splitting optimization method was therefore tested on the real world *CityBlocks* dataset. However, the proposed method is outlier robust only up to an extent—it is based on the assumption that the majority of tracks that were split up have been triangulated correctly. To remove those outlier track correspondences, the following criterion can be used: If two

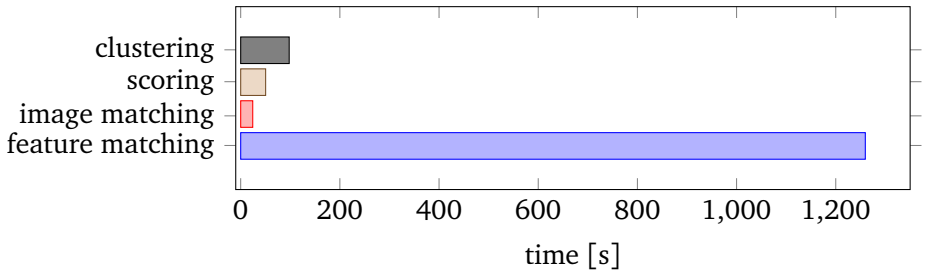

Figure 6.3: Comparison of the single parts of the vocabulary tree guided matching pipeline. The operations on the tree itself take only a fraction of the time compared to the feature matching.

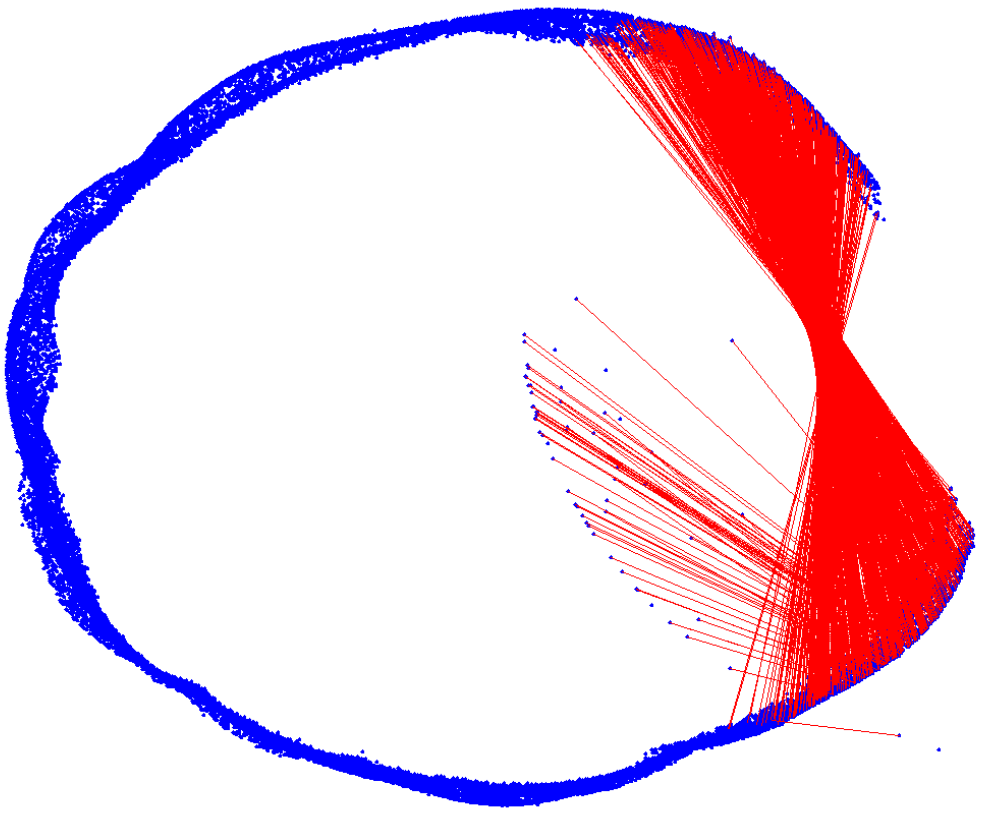

Figure 6.4: Top view of the modified *Cylinder* dataset. The blue points depict the track positions, the red lines connect the tracks that were split up and should be the same position.

consecutive frames have a camera pose with a distance substantially larger than the mean distance of all consecutive camera poses, then the track splitting information is kept, otherwise it is discarded. The tracks and the matches are visualized before and after this filtering step in Figure 6.5.

Figure 6.6 and Figure 6.7 show a comparison of the reconstructed geometry and camera path before and after the global optimization using the loop closing constraints. The constrained optimization could correct for the misalignment errors up to a certain extent.

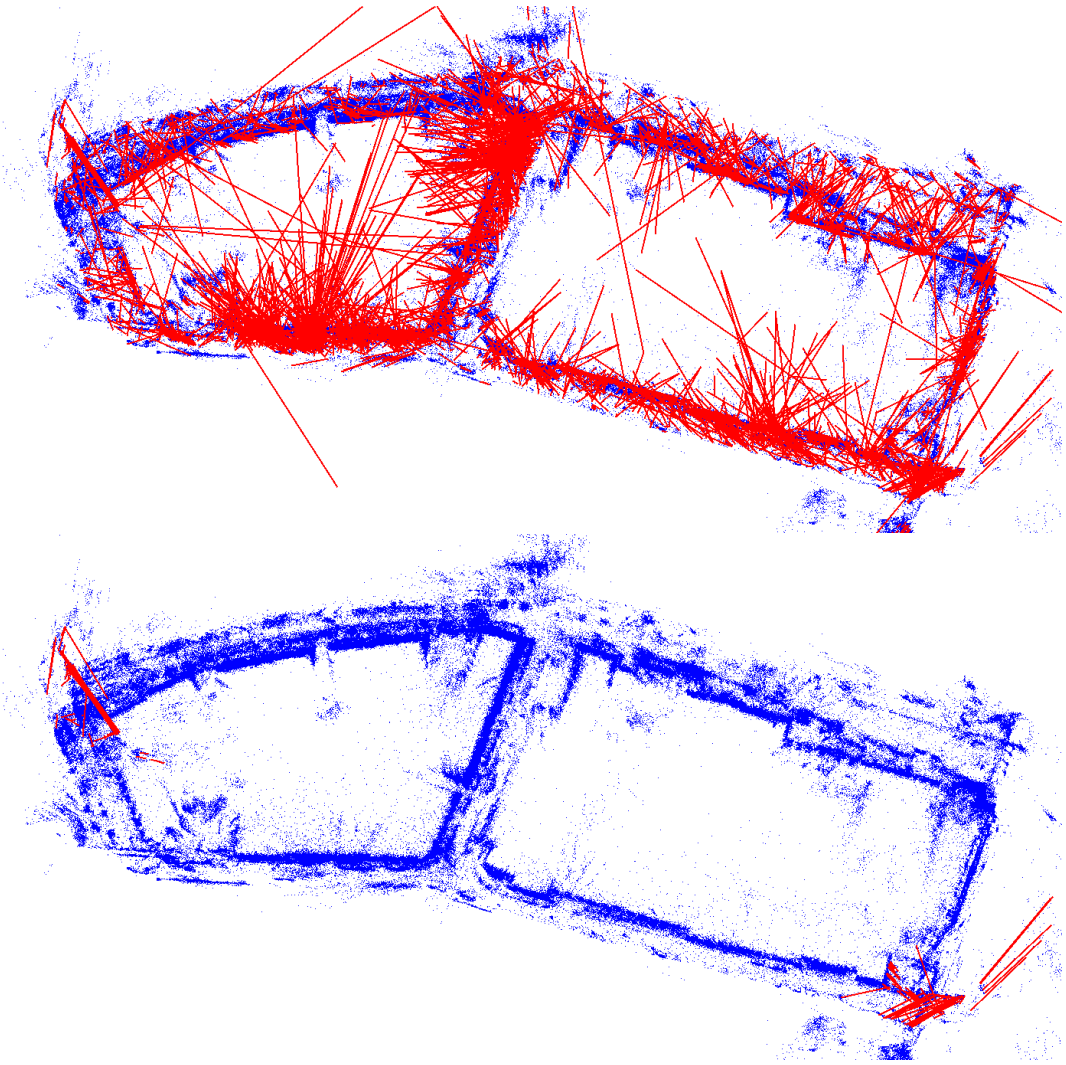

Figure 6.5: A visualization of the drift error detection using track splitting. The two graphics show a top view of the 3D points (blue) of the *CityBlocks* dataset. Red lines show correspondences between tracks that have been split. *Top:* Even correctly reconstructed parts of the scene contain track matches because there are lots of outliers. *Bottom:* The filtered matches using a distance threshold that removes matches if the associates camera poses are close and therefore correct. These matches can then be used as loop constraints.

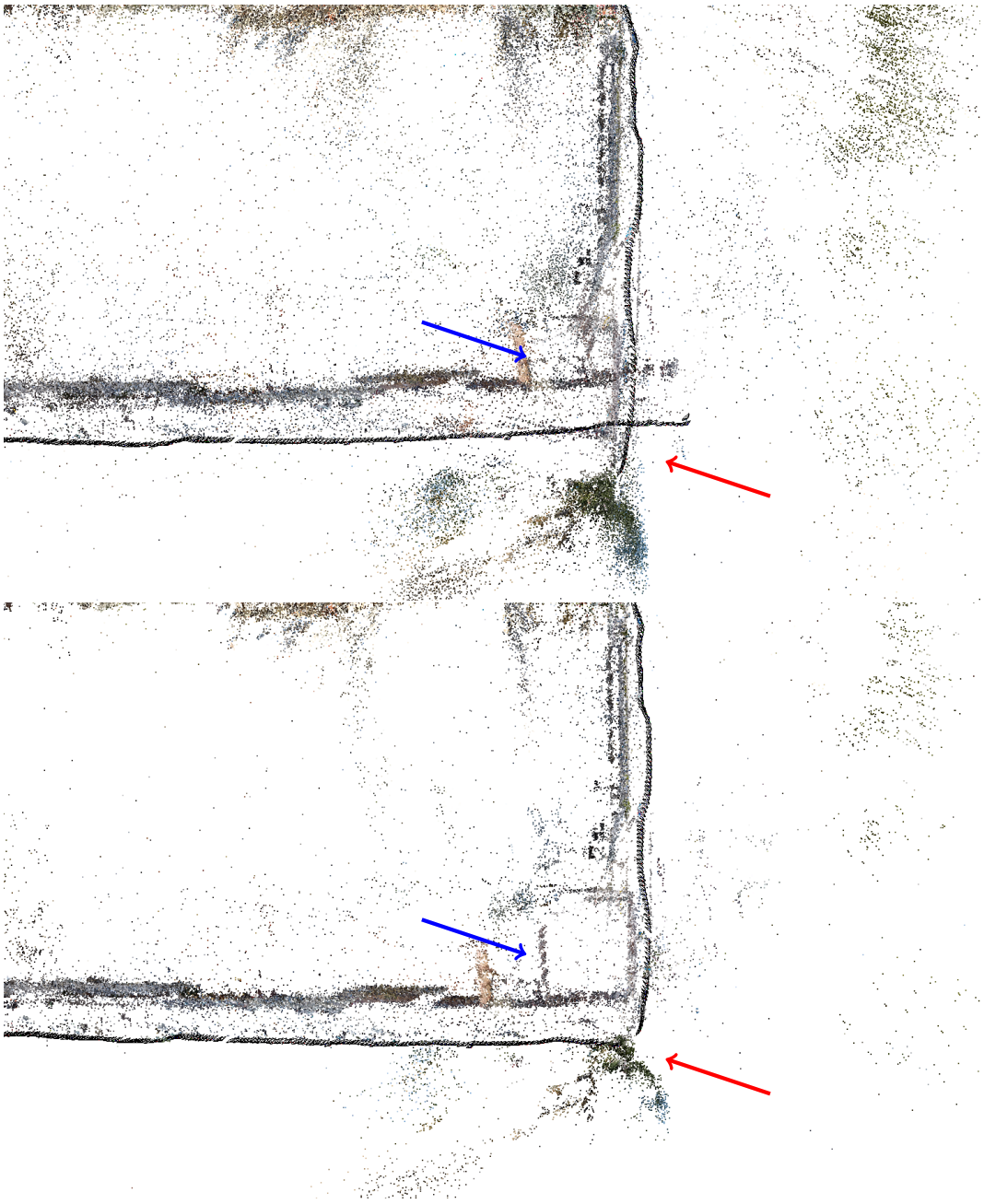

Figure 6.6: *Top:* Top view of the reconstructed *CityBlocks* dataset after bundle adjustment. The geometry of the corner house is overlapped (blue) as well as the camera trajectory (red). *Bottom:* After the global optimization using loop closure constraints the problematic areas are fixed.

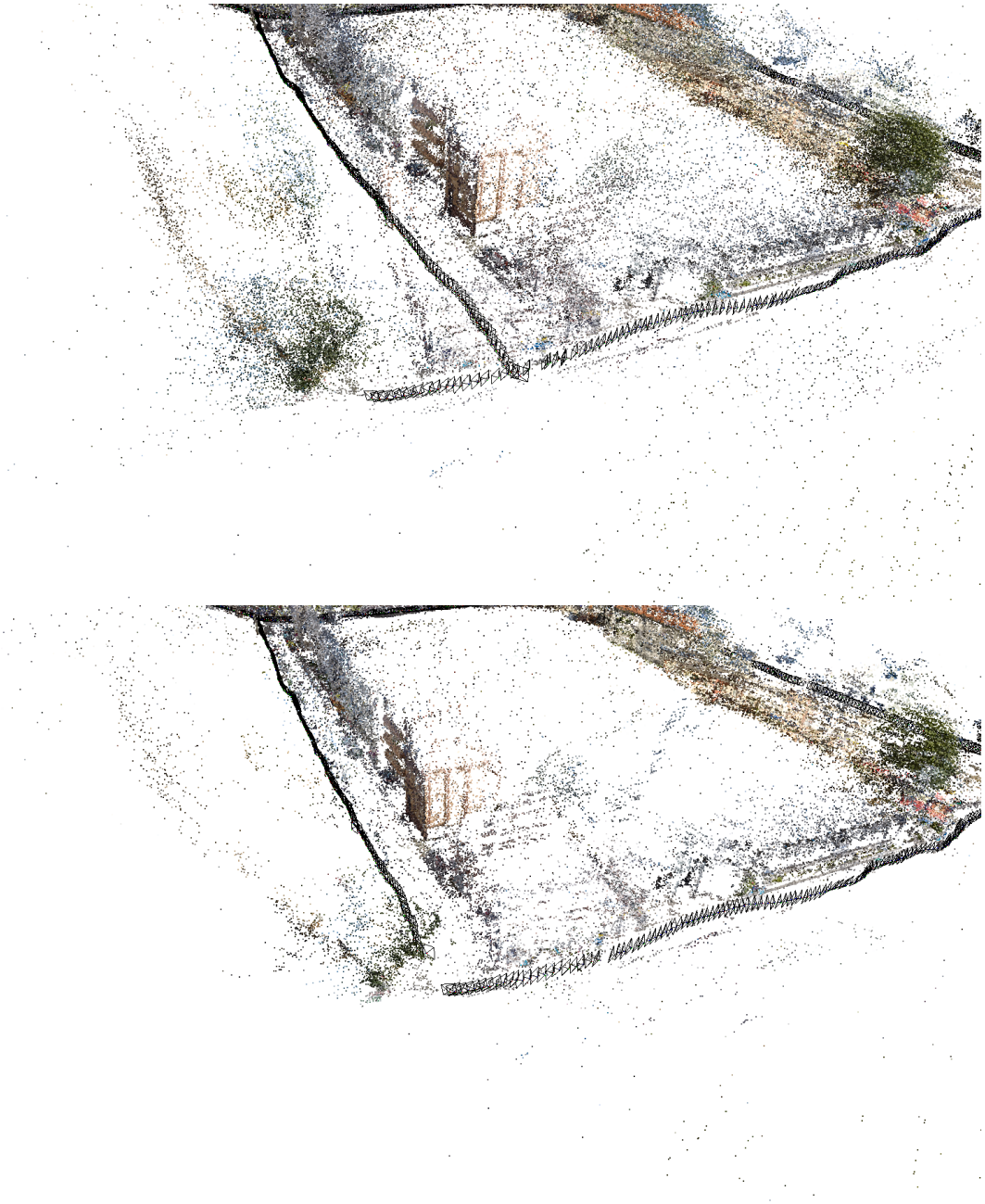

Figure 6.7: *Top:* Side view of the reconstructed *CityBlocks* dataset after bundle adjustment. *Bottom:* After the global optimization using loop closure constraints the camera path alignment is improved. Nevertheless the camera trajectory is still not perfectly closed. This is a limitation of the current approach and due to the camera positions being optimized only indirectly.

7 Conclusion and Future Work

During the work on this thesis, bundling software was developed that is capable of reconstructing scenes captured using long outdoor video sequences.

7.1 Summary

The matching including the detection of possible loops in the camera path or captured scene geometry is implemented efficiently using a vocabulary tree approach which outputs putative matching image pairs. The heuristic basically returns only relevant image pair matches to speed up the feature matching process. This approach produces robust results on several tested scenes. Compared to an exhaustive matching, the vocabulary tree approach is up to 23 times faster on an real world data set consisting of 800 images. It is even 3 times faster than the low-resolution matching approach used by Fuhrmann et al. [2014].

The SfM pipeline with accelerated matching worked well on small and dense scenes, but larger outdoor scenes could not be reconstructed correctly due to camera misalignments and drift. An approach using track splitting was presented and tested which was able to correct for these errors on a large real world dataset.

7.2 Global SfM

However, after preliminary test, a *global* Structure from Motion algorithms is a more promising approach to estimate camera poses from a video sequence. The reason is that the errors are spread evenly over the whole dataset without running into local misalignments.

Additionally, It is easier to formulate and solve an optimization problem if the camera positions can be optimized directly. In contrast to incremental Structure from Motion algorithms, global approaches only use relative pose information. But without a globally consistent model which allows for easier removal of outliers as in the incremental case,

the outlier removal is difficult. Wilson and Snavely [2014] use local two-view translation directions and project these vectors along various directions to create one dimensional subproblems of the type *minimum feedback arc set*. Using this approach, outliers can be removed without knowing a globally consistent model. The two-view rotations and translations are then used as input to a global optimization: The error function can be formulated as

$$E_r = \sum_{(i,j)\in E} \left\| \hat{t}_{ij} - \frac{t_j - t_i}{\|t_j - t_i\|} \right\|_2^2 \tag{7.1}$$

where $\hat{t}_{ij}$ is the relative two-view translation from the two-view pose estimation. Hence the scale is not known and thus the difference of the global translations t_i and t_j is normalized. These global translations are the parameters that get modified during the optimization.

Wilson and Snavely [2014] formulated this optimization problem using the Ceres Solver as well, so the loop closure optimization code could be adapted to use this global SfM approach.

Bibliography

Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., and Szeliski, R. (2011). Building rome in a day. *Communications of the ACM*, 54(10):105–112.

Agarwal, S. and Mierle, K. (2012). Ceres Solver: Tutorial & Reference. *Google Inc*.

Akbarzadeh, A., Frahm, J.-M., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Merrell, P., Phelps, M., Sinha, S. N., Talton, B., 0002, L. W., Yang, Q., Stewenius, H., Yang, R., Welch, G., Towles, H., Nister, D., and Pollefeys, M. (2006). Towards Urban 3D Reconstruction from Video. In *3DPVT*, pages 1–8.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer vision and image understanding*, 110(3):346–359.

Blender Foundation (2014). *Blender - a 3D modelling and rendering package*. Blender Institute, Amsterdam.

Clemente, L. A., Davison, A. J., Reid, I. D., Neira, J., and Tardós, J. D. (2007). Mapping Large Loops with a Single Hand-Held Camera. In *Robotics: Science and Systems*.

Cummins, M. and Newman, P. (2008). FAB-MAP: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6):647–665.

Engel, J., Schöps, T., and Cremers, D. (2014). LSD-SLAM: Large-Scale Direct Monocular SLAM. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, volume 8690, pages 834–849. Springer International Publishing.

Frahm, J.-m., Pollefeys, M., Lazebnik, S., Gallup, D., Clipp, B., Raguram, R., Wu, C., Zach, C., and Johnson, T. (2010). Fast robust large-scale mapping from video and internet photo collections. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):538–549.

Fuhrmann, S., Langguth, F., and Goesele, M. (2014). MVE - A Multi-View Reconstruction Environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*.

Grauman, K. and Darrell, T. (2007). Approximate Correspondences in High Dimensions. In Scholkopf, B., Platt, J. C., and Hofmann, T., editors, *Advances in Neural Information Processing Systems 19*, Cambridge, MA. MIT Press.

Hartley, R. I. and Dano, N. Y. (2000). Reconstruction from six-point sequences. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 480–486.

Klingner, B. M., Martin, D., and Roseborough, J. (2013). Street View Motion-from-Structure-from-Motion. In *ICCV*, pages 953–960.

Klopschitz, M., Zach, C., Irschara, A., Schmalstieg, D., and Hill, C. (2008). Generalized Detection and Merging of Loop Closures for Video Sequences. *Proceedings of 3DPVT'08 - the Fourth International Symposium on 3D Data Processing, Visualization and Transmission Generalized*.

Kneip, L., Scaramuzza, D., and Siegwart, R. (2011). A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2969–2976.

Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g^2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

Moulon, P., Monasse, P., and Marlet, R. (2013). Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion. In *ICCV*, pages 3248–3255. IEEE.

Muja, M. and Lowe, D. G. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press.

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2:2161–2168.

Perlin, K. (2002). Improving noise. In *ACM Transactions on Graphics (TOG)*, volume 21, pages 681–682.

Scaramuzza, D., Fraundorfer, F., and Pollefeys, M. (2010). Closing the Loop in Appearance-guided Omnidirectional Visual Odometry by Using Vocabulary Trees. *Robot. Auton. Syst.*, 58(6):820–827.

Scaramuzza, D., Martinelli, A., and Siegwart, R. (2006). A toolbox for easily calibrating omnidirectional cameras. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 5695–5701.

Umeyama, S. (1991). Least-Squares Estimation of Transformation Parameters Between Two Point Patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):376–380.

Williams, B., Cummins, M., Neira, J., and Newman, P. (2009). A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2008). An image-to-map loop closing method for monocular SLAM. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2053–2059.

Wilson, K. and Snavely, N. (2014). Robust Global Translations with 1DSfM. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, volume 8691, chapter Lecture Notes in Computer Science, pages 61–75. Springer International Publishing.

Wu, C., Agarwal, S., Curless, B., and Seitz, S. M. (2011). Multicore bundle adjustment. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3057–3064.